Your search matched **0** of **1027552** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance Descending** order.

**Refine This Search:**
You may refine your search by editing the current search expression or enter new one in the text box.

mutex <near> frequency Search

☐ Check to search within this result set

**Results Key:**
**JNL** = Journal or Magazine **CNF** = Conference **STD** = Standard

---

**Results:**
**No documents matched your query.**

searched mutex (near) frequency got 5,105 hits -?

Term used mutex Found 384 of 131,734

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Optimal Mutex policy in Ada 95

Jim Abu-Ras

December 1995 **ACM SIGAda Ada Letters**, Volume XV Issue 6

Full text available: pdf(570.43 KB) Additional Information: full citation, abstract, index terms

Priority inversion is any situation where low priority tasks are served before higher priority tasks. It is recognized as a serious problem for Real-Time Systems. In this paper, we present some of the important features of the Real-Time Annex of Ada 95. We also implement the *Optimal Mutex Policy (OMP)* in Ada 95 to better illustrate Ada's new usefulness for Real-Time programming. A detailed discussion of this protocol and other related issues are presented.

2 Object-oriented programming for embedded systems

Stuart Maclean, Sean Smith

September 1995 **ACM SIGPLAN Notices**, Volume 30 Issue 9

Full text available: pdf(552.06 KB) Additional Information: full citation, abstract, citings, index terms

Embedded systems are hard to program. Much of the effort is concentrated on processor-management code, effort totally unrelated to the application being built. Concurrent programming in embedded systems introduces even more problems --- mechanisms are needed to protect critical data areas. The application of object modeling to such mechanisms results in numerous benefits: fewer programming errors, enhanced portability and improved reuse. We present some simple in techniques for realising these b ...

3 Implementation of resilient, atomic data types

William Weihl, Barbara Liskov

April 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 2

Full text available: pdf(2.19 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A major issue in many applications is how to preserve the consistency of data in the presence of concurrency and hardware failures. We suggest addressing this problem by implementing applications in terms of abstract data types with two properties: Their objects are atomic (they provide serializability and recoverability for activities using them) and resilient (they survive hardware failures with acceptably high probability). We define what it means for abstract data types to be atomic and ...

4 Implementing Ada 9X features using POSIX Threads: design issues
E. W. Giering, Frank Mueller, T. P. Baker
October 1993 **Proceedings of the conference on TRI-Ada '93**
Full text available: pdf(1.49 MB) Additional Information: full citation, references, citings, index terms

5 Debuggable concurrency extensions for standard ML
Andrew P. Tolmach, Andrew W. Appel
December 1991 **ACM SIGPLAN Notices , Proceedings of the 1991 ACM/ONR workshop on Parallel and distributed debugging**, Volume 26 Issue 12
Full text available: pdf(1.22 MB) Additional Information: full citation, references, citings, index terms

6 Specification and implementation of resilient, atomic data types
William Weihl, Barbara Liskov
June 1983 **Proceedings of the 1983 ACM SIGPLAN symposium on Programming language issues in software systems**
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

A major issue in many applications is how to preserve the consistency of data in the presence of concurrency and hardware failures. We suggest addressing this problem by implementing applications in terms of abstract data types with two properties: Their objects are atomic (they provide serializability and recoverability for activities using them) and resilient (they survive hardware failures with acceptably high probability). We define what it means for abstract data types to be atomic and ...

7 Linguistic support for atomic data types
William E. Weihl
April 1990 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 12 Issue 2
Full text available: pdf(2.10 MB) Additional Information: full citation, abstract, references, index terms

The problems of concurrency and failures in distributed systems can be addressed by implementing applications in terms of atomic data types: data types whose objects provide serializability and recoverability for transactions using them. The specifications of the types can be used to permit high levels of concurrency among transactions while still ensuring atomicity. However, highly concurrent implementations can be quite complicated. In this paper we analyze the expressive power of existin ...

8 Perfecting preemption threshold scheduling for object-oriented real-time system design: from the perspective of real-time synchronization
Saehwa Kim, Seongsoo Hong, Tae-Hyung Kim
June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems**, Volume 37 Issue 7
Full text available: pdf(380.92 KB) Additional Information: full citation, abstract, references, index terms

In spite of the proliferation of object-oriented design methodologies in contemporary software development, their application to real-time embedded systems has been limited because of the practitioner's conservative attitude toward handling timing constraints. In fact, this conservative attitude is well-grounded because traditional priority-based scheduling techniques cannot be straightforwardly integrated into them. The automated implementation from the object-oriented real-time designs usually ...

**Keywords:** object-oriented real-time system design, preemption threshold scheduling, priority ceiling protocol, priority inheritance protocols, real-time synchronization

**9** Using POSIX threads to implement Ada tasking: description of work in progress
E. W. Giering, T. P. Baker
December 1992 **Proceedings of the conference on TRI-Ada '92**
Full text available: pdf(1.22 MB) Additional Information: full citation, references, citings, index terms

**10** A partially deadlock-free typed process calculus
Naoki Kobayashi
March 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 2
Full text available: pdf(562.16 KB) Additional Information: full citation, references, citings, index terms

**Keywords:** concurrency, deadlock-freedom, type theory

**11** Application-defined scheduling in Ada
Mario Aldea Rivas, Michael González Harbour
April 2002 **ACM SIGAda Ada Letters , Proceedings of the 11th international workshop on Real-time Ada workshop**, Volume XXII Issue 4
Full text available: pdf(147.48 KB) Additional Information: full citation, abstract, references

This paper presents an application program interface (API) that enables Ada applications to use application-defined scheduling algorithms in a way compatible with the scheduling model of the Ada 95 Real-Time Systems Annex. Several application-defined schedulers, implemented as special user tasks, can coexist in the system in a predictable way. This API is currently implemented on our operating system MaRTE OS.

**Keywords:** POSIX, ada 95, kernel, operating systems, real-time systems, scheduling

**12** Analyzing synchronization problems by using event histories as languages
Robert D. Dixon, David Hemmendinger
February 1986 **Proceedings of the 1986 ACM fourteenth annual conference on Computer science**
Full text available: pdf(588.74 KB) Additional Information: full citation, references

**13** What is Multi-Threading?
Martin McCarthy
February 1997 **Linux Journal**
Full text available: html(25.02 KB) Additional Information: full citation, abstract, citings, index terms

A primer on multi-threading: the process whereby linux manages several tasks simultaneously

**14** Synchronization primitives for a multiprocessor: a formal specification
A. Birrell, J. Guttag, J. Horning, R. Levin
November 1987 **ACM SIGOPS Operating Systems Review , Proceedings of the eleventh**

**ACM Symposium on Operating systems principles**, Volume 21 Issue 5

Full text available: pdf(812.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Formal specifications of operating system interfaces can be a useful part of their documentation. We illustrate this by documenting the Threads synchronization primitives of the Taos operating system. We start with an informal description, present a way to formally specify interfaces in concurrent systems, give a formal specification of the synchronization primitives, briefly discuss the implementation, and conclude with a discussion of what we have learned from using the specification for ...

**15** A language with distributed scope

Luca Cardelli

January 1995 **Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

Obliq is a lexically-scoped, untyped, interpreted language that supports distributed object-oriented computation. Obliq objects have state and are local to a site. Obliq computations can roam over the network, while maintaining network connections. Distributed lexical scoping is the key mechanism for managing distributed computation.

**16** Waiting algorithms for synchronization in large-scale multiprocessors

Beng-Hong Lim, Anant Agarwal

August 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 3

Full text available: pdf(2.72 MB) Additional Information: full citation, abstract, references, citings, index terms

Through analysis and experiments, this paper investigates two-phase waiting algorithms to minimize the cost of waiting for synchronization in large-scale multiprocessors. In a two-phase algorithm, a thread first waits by polling a synchronization variable. If the cost of polling reaches a limit Lpoll and further waiting is necessary, the thread is blocked, incurring an additional fixed cost, B. The choice of Lpoll

**Keywords:** barriers, blocking, competitive analysis, locks, producer-consumer synchronization, spinning, waiting time

**17** Interrupts as threads

Steve Kleiman, Joe Eykholt

April 1995 **ACM SIGOPS Operating Systems Review**, Volume 29 Issue 2

Full text available: pdf(492.24 KB) Additional Information: full citation, abstract, citings, index terms

Most operating system implementations contain two fundamental forms of asynchrony; processes (or equivalently, internal threads) and interrupts. Processes (or threads) synchronize using primitives such as mutexes and condition variables, while interrupts are synchronized by preventing their occurrence for a period of time. The latter technique not only is expensive, but it locks out interrupts on the possibility that an interrupt will occur and interfere with the particular critical section of c ...

**18** Performance measurements for multithreaded programs

Minwen Ji, Edward W. Felten, Kai Li

June 1998 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems**, Volume 26 Issue 1

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded programming is an effective way to exploit concurrency, but it is difficult to debug and tune a highly threaded program. This paper describes a performance tool called Tmon for monitoring, analyzing and tuning the performance of multithreaded programs. The performance tool has two novel features: it uses "thread waiting time" as a measure and constructs thread waiting graphs to show thread dependencies and thus performance bottlenecks, and it identifies "semi-busy-waiting" points w ...

**19** POSIX thread libraries
Felix Garcia, Javier Fernandez
February 2000 **Linux Journal**

Full text available: html(20.36 KB) Additional Information: full citation, abstract, references, index terms

The authors have studied five

**20** Introduction to Multi-Threaded Programming
Brian Masney
May 1999 **Linux Journal**

Full text available: html(10.32 KB) Additional Information: full citation, abstract, citings, index terms

A description of thread programming basics for C programmers

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next